System, Method, and Computer Program Product For In-Place, Lightweight Ack Promotion In A Cable Modem Environment

Inventors: David Pullen
Rick Pitchford
Dannie Gay
John Horton

Cross-Reference To Related Applications

[0001] This application claims the benefit of U.S. Provisional Application 60/464,956, entitled "System and Method for In-Place, Lightweight Ack Packet Promotion," filed April 24, 2003, which is incorporated by reference herein in its entirety.

Background Of The Invention

Field of the Invention

[0002] The present invention is related to improving the performance of Transmission Control Protocol (TCP) based applications on Data Over Cable Service Interface Specification (DOCSIS) channels.

Related Art

[0003] The importance to the modern economy of rapid data access and exchange cannot be overstated. This explains the exponentially increasing popularity of the data access and exchange via cable networks (including coaxial cable or Hybrid fiber coaxial cable), the Internet, intranets, wireless networks, satellites and so forth (i.e., communication mediums). Rapid data access and exchange is partly dependent upon how efficiently bandwidth is allocated to a data provider in order for the data provider to transfer the requested data to a user via one of the communication mediums mentioned above.

[0004] One very desirable solution for rapid data access and exchange is via cable networks and cable modems. Cable modems provide communications

on cable networks. In general, a user connects a cable modem to the TV outlet for his or her cable TV, and the cable TV operator connects a cable modem termination system (CMTS) in the operator headend. The CMTS is a central device for connecting the cable network to a data network like the Internet. The CMTS is a central distribution point for a cable system. Data flows downstream from the CMTS to the cable modem (i.e., downstream communication). Alternatively, data flows upstream from the cable modem to the CMTS (i.e., upstream communication).

[0005] A common cable modem standard today is the CableLabs® Certified™ Cable Modem project, also known as DOCSIS® (Data Over Cable Service Interface Specification), which defines interface requirements for cable modems involved in high-speed data distribution over cable television system networks. The DOCSIS specifications define technical requirements for both cable modems and CMTS.

[0006] The performance of Transmission Control Protocol (TCP) based applications (such as File Transfer Protocol (FTP) and HyperText Transfer Protocol (HTTP)) is severely limited on DOCSIS channels due to the extreme asymmetry between the speed of the upstream and downstream paths, as well as due to the high latency of the upstream path relative to the lower latency of the downstream path. These issues also apply to numerous other physical link types (satellite, DSL, etc). One of the approaches to alleviating the bottlenecks is called Ack Filtering, where the customer premises equipment ("CPE") device (such as, a cable modem or router) looks for TCP Ack packets being queued to be transmitted upstream, and removes "older" Ack packets in favor of sending the newer Ack packet. The older Ack packet is removed only if the packet contains no data. If there is data in the older Ack packet, then it cannot be removed without discarding the data. However, the newer Ack number can be promoted to the older packet. A TCP Ack is any TCP packet that has the Ack bit set in the flags portion of the TCP header, and the Acknowledgement Number value indicates the Sequence Number being acknowledged.

[0007] There are several issues related to Ack Filtering that can make the implementation difficult and inefficient, especially when taking into account DOCSIS features like piggyback requests and concatenation.

Summary Of The Invention

[0008] A method, system, and computer program product for in-place, lightweight Ack packet promotion is provided. The method includes receiving a new Ack packet; searching through a transmit queue for an old Ack packet that corresponds to the new Ack packet; and replacing the data in a number field, a checksum field, a window size field, and/or a timestamp options (TSO) field of the old Ack packet with data in a number field, a checksum field, a window size field, and/or a TSO field of the new Ack packet.

Brief Description Of The Figures

[0009] The present invention will be described with reference to the accompanying drawings, wherein:

[0010] FIG. 1 is a block diagram representing an example operating environment of the present invention according to an embodiment.

[0011] FIG. 2 represents a TCP Ack packet according to an embodiment of the present invention.

[0012] FIG. 3 illustrates the high level operation of the invention.

[0013] FIG. 4 represents a TCP Ack packet according to another embodiment of the present invention.

Detailed Description Of The Preferred Embodiments

A. Overview of the Invention

[0014] As mentioned above, there are several issues related to Ack Filtering that can make the implementation difficult and inefficient, especially when

taking into account DOCSIS features like piggyback requests and concatenation. Ack Filtering involves removing "unnecessary" TCP Ack packets from the outbound or transmit queue(s). Because TCP Acks are cumulative, it is not necessary to transmit all of them. For example, if a client tries to send three Ack packets, and Ack 3 arrives before Ack 1 has been transmitted, it is possible to discard Acks 1 and 2 and only send Ack 3. This is because Ack 3 includes the acknowledgement for data that would have been provided by Ack 1 and Ack 2.

[0015] One issue with many hardware direct memory access ("DMA") engines is that once a packet is queued for transmission, the hardware will fetch the buffer (or buffer descriptors), making it impossible to remove that packet from the queue without significantly interrupting the transmission service. Additionally, it may not be possible to know whether or not the packet at the front of the queue is currently being transmitted. Here, removing the packet while it is being transmitted may cause a number of problems, including but not limited to, packet corruption, loss of hardware state synchronization, etc. One solution to this is to do Ack Filtering only on packets beyond the one at the head of the queue. However, this introduces latency to the system that should be avoided.

[0016] In order to provide optimal operation, and to ensure that the most recent Ack is transmitted as soon as possible, the present invention provides in-place, lightweight Ack promotion. Rather than removing the old Ack packet from the queue and putting the new packet in the queue, the present invention copies only the necessary fields from the new Ack to the buffer containing the old Ack. The present invention then discards the packet containing the new Ack. This essentially promotes the old Ack to contain the acknowledgement information from the new Ack, making the new Ack packet unnecessary. This will be described further below with reference to FIGs. 1-3.

[0017] For illustration purposes, the present invention is described in terms of being utilized with a cable network and DOCSIS. It should be understood that the present invention is independent of the actual physical layer of

transmission utilized by DOCSIS (e.g., TDMA, SCDMA, etc.). It should also be understood that the present invention is not limited to use with a cable network and/or DOCSIS. In fact, the present invention may be used with any communication medium, including but not limited to, Ethernet-based networks (e.g., IEEE 802.3), fiber optic networks, wireless networks and satellite-based networks.

B. System Architecture Overview

[0018] FIG. 1 is a block diagram representing an example operating environment of the present invention. It should be understood that the example operating environment in FIG. 1 is shown for illustrative purposes only and does not limit the invention. Other implementations of the operating environment described herein will be apparent to persons skilled in the relevant art(s) based on the teachings contained herein, and the invention is directed to such other implementations. Referring to FIG. 1, a CMTS 102, a cable modem 104, downstream communication 106 and upstream communication 108, are shown. CMTS 102 further includes a CMTS scheduler 110. Cable modem 104 includes cable modem Media Access Controller (MAC) 114 and one or more transmit queues 112. It is noted that communication lines 106 and 108 may be the same line or separate lines. Each of the components in FIG. 1 will be briefly described next.

[0019] In general, cable modem 104 forwards or provides data via communications on cable networks. Cable modem 104 receives data from a user that needs to be transferred via a cable network. In order to do this, cable modem 104 requests that CMTS 102 grant to it the necessary bandwidth via a request packet. It is noted that only one cable modem 104 is shown. However, it is contemplated that there may be dozens, or hundreds, or thousands of cable modems in a system.

[0020] As mentioned, cable modem 104 receives data from a user to be transferred via a cable network. In addition, CMTS 102 receives data from the network to be transferred to cable modem 104.

[0021] Cable modem MAC 114 is coupled to cable modem 104. In general, cable modem MAC 114 is responsible for multiplexing the internal traffic, (i.e., requesting the necessary bandwidth that cable modem 104 needs to transfer its current types of data). Cable modem MAC 114 sends bandwidth requests to CMTS 102 via upstream communication 108. At a high level, each bandwidth request may include two fields. The two fields include a data service identifier (SID) and the amount of required bandwidth.

[0022] Transmit queue(s) 112 is used by cable modem 104 to queue packets to be transmitted via upstream communication 108. One type of packet stored in transmit queue(s) 112 is TCP Ack packets. TCP Ack packets are described further below with reference to FIG. 2 and FIG. 4.

[0023] CMTS 102 is a central device for connecting the cable network to a data network. CMTS scheduler 110 is a bandwidth manager. CMTS scheduler 110, as a bandwidth manager, decides how to grant available bandwidth according to the current bandwidth requests. This grant is done via a packet sent onto downstream communication 106 to cable modem 104. It is important to note that this request/grant negotiation between cable modem 104 and CMTS 102 is one of the reasons why the upstream latency is high.

[0024] Another reason for the high latency is the MAP lead time. CMTS 102 sends bandwidth grants downstream to cable modem 104 a certain amount of time before the bandwidth is to be used. This ensures that the grants will not arrive too late for cable modem 104 to make use of them. The amount of lead time is often configurable, and can range from less than one millisecond to more than ten milliseconds.

[0025] Another reason for latency is the downstream interleaver depth. The interleaver provides some protection against errors on the downstream 106. There exists a direct correlation between interleaver depth and the degree of protection and/or latency. If more depth is provided, greater protection and higher latency can be achieved. The parameter for interleaver depth is also configurable. Valid depths are specified in the DOCSIS RFI specifications.

The present invention overcomes all of these forms of latency in a DOCSIS-compliant system.

[0026] TCP Ack packets are described further with reference to FIG. 2. Note that a TCP Ack packet is any TCP packet with the Ack bit set. FIG. 2 illustrates an example TCP Ack packet 202. Among other fields, packet 202 includes a TCP header field 204. Header field 204 includes a 32-bit Ack number field 206 and a 16-bit TCP checksum field 208. ACK number field 206 is located 8 bytes from the start of header field 204. TCP checksum field 208 is located 16 bytes from the start of header field 204.

[0027] FIG. 4 illustrates another example of TCP Ack packet 202, which shows an entire packet. An LLC header 402, an IP header 404, payload 406, and CRC field 408 are included. TCP header field 204 includes information for handling the window size and timestamp options (TSO) in additions to fields for ACK number and checksum. Other fields can be included with packet 202 as described in the TCP DARPA Internet Program Protocol Specification (RFC 793), which is incorporated herein by reference.

[0028] As stated above, the present invention ensures that the most recent Ack packet is transmitted as soon as possible by implementing in-place, lightweight Ack promotion. Rather than removing the old Ack packet from transmit queue(s) 112 and putting the new packet in transmit queue(s) 112, the present invention copies only the necessary fields from the new Ack packet to the buffer containing the old Ack packet. The present invention then discards the Ack packet containing the new Ack. This essentially promotes the old Ack packet to contain the acknowledgment information from the new Ack packet, making the new Ack packet unnecessary. These necessary fields include Ack number field 206 and TCP checksum field 208, as shown in FIG. 2, as well as the window size field and TSO field, as shown in FIG. 4.

[0029] Note that using in-place, lightweight Ack promotion ensures that the most recent Ack packet will be transmitted. In addition, because the Ack number field 206 and checksum field 208 (as well as the window size field

and TSO field, if included) are near the end of the packet 202, the present invention can even promote Ack packets while they are being transmitted.

[0030] This approach also has a benefit related to rate shaping. Cable modems are required to implement rate shaping to limit the amount of bandwidth that they use over a period of time. This delays packets that would exceed the rate limit. Because the in-place Ack promotion code causes the new Ack packet to be discarded, it does not count against rate limits, and thus allows more packets to be sent sooner.

C. Operation of the Present Invention

[0031] FIG. 3 is a flowchart that illustrates the operation of the present invention. FIG. 3 starts at step 302, where cable modem 104 receives a new Ack packet.

[0032] In step 304, cable modem 104 searches through transmit queue(s) 112 to determine whether queue(s) 112 contains an old Ack packet that corresponds to the new Ack packet. The present invention must ensure that it replaces data in an old Ack packet that corresponds to the same Ack stream as the new Ack packet.

[0033] Each client (e.g. a PC) can communicate with multiple TCP streams, and there may be multiple clients attached to cable modem 104. Each of the streams is identified by the tuple (source IP address, destination IP address, source port number, destination port number). If the present invention were to replace the Ack packet information for stream 1 with an Ack packet for stream 2, it would cause the remote host for stream 1 to receive incorrect Ack information, possibly causing the transfer to fail, or causing unexpected retransmission. Additionally, the remote host for stream 2 would not receive an Ack packet at all, which would cause a pause in the data flow or retransmission of data.

[0034] Even within a single Ack stream, a new Ack packet can only replace the information from a nearly identical old Ack packet. The Ack packets can be used to carry other information (TCP options, payload, etc), and the present

invention cannot allow a new Ack packet to cause this additional information to be changed, corrupted, or lost. In the same vein, the present invention cannot allow extra information carried by the new Ack packet to be lost. As a result, when searching through the queue of packets in transmit queue(s) 112, cable modem 104 must only select an old Ack packet for replacement if it matches the new Ack packet in every way except for data in the acknowledgment number field (i.e., Ack number field 206), window size field, and TSO field. This presents a performance problem because of the amount of data that must be compared (28 bytes or more) in order to ensure that the old and new Ack packets belong to the same stream and that the old and new Ack packets are nearly identical. In many cases, the present invention is attempting to promote the new Ack information to the buffer that is at the head of transmit queue(s) 112. The present invention needs to be able to identify the similar Ack packets quickly so that the promotion can occur before the old Ack packet is transmitted.

[0035] The present invention hashes the information from the old and new Ack packets down to a reasonably unique and small piece of information (such as a 32-bit stream id number that can be compared in a single CPU cycle) that is used to quickly identify whether or not the old and new Ack packets belong to the same stream, and whether or not there is additional data present that would prevent the new Ack packet from being promoted. In particular, the present invention utilizes the TCP checksum field 208 (FIG. 2 or FIG. 4) to act as a hash algorithm. The TCP checksum field 208 includes not only the entire TCP header and payload data, but also a pseudo-header that contains the IP addresses, IP protocol type, and the length of the TCP data. Other information and requirements are described in the TCP DARPA Internet Program Protocol Specification (RFC 793). Thus the present invention only needs to remove the acknowledgment number, window size, and TSO from the checksum, and the remaining value will be identical for Ack packets that differ only by the ack number.

[0036] Note, however, that the 16-bit checksum is not a perfect hash algorithm; for any two arbitrary packets, there is a 1 in 65,535 chance that they will produce the same checksum value. In order to make the hash value more robust, the present invention combines the modified checksum with the 16-bit source port number from the TCP header, giving a 1 in 4.29 billion chance that the hash will be the same. The source port number is chosen uniquely for every socket within a client that is communicating with a server, and it is very unlikely that two different clients will choose the same source port number at the same time. But, even if they did, the two clients must have different IP addresses, which will make the checksum produce a different value.

[0037] When the new Ack packet is received from the client, cable modem 104 calculates this 32-bit stream id, and stores it along with the Ack packet as it is queued to be transmitted. Thus, the present invention only calculates the stream id once per Ack packet. When a new Ack packet arrives, the present invention calculates the stream id and uses it to search the list of queued packets in transmit queue(s) 112, comparing the stream ids. If a match is found, the present invention promotes the Ack packet.

[0038] Thus, in step 306, if a corresponding old Ack packet is found, then cable modem 104 replaces the data in the Ack number field, the checksum field, window size field, and TSO field in the buffer containing the old Ack packet with the data found in the same fields of the new Ack packet. The new Ack packet is then discarded.

D. Robust Ack Transmission

[0039] Because the present invention often attempts to do in-place Ack promotion on the Ack packet that is at the front of transmit queue(s) 112, there is a risk that the Ack packet might be transmitted before it is modified. If the hardware is reading the portion of the Ack packet that is before the Ack number field, the present invention's promotion will work and the new Ack packet will be sent.

[0040] If the hardware is reading the portion of the Ack packet that is after the TCP checksum field 208, the present invention's in-place modification will be ignored and an older Ack packet will be sent.

[0041] If the hardware is reading the portion of the Ack packet between the Ack number field 206 and the checksum field 208, the Ack packet can be corrupted (the checksum will not match the information that is in the header, and the remote system will discard it). However, in an embodiment of the present invention, a Data Cache is utilized to minimize the risk of this type of corruption. The Data Cache is built into the processor to store the updated information. After the new Ack information has been completely copied to the Data Cache, it is then flushed to physical memory (e.g., RAM) so that a DMA controller can read it when the packet is transmitted. The data that is updated spans two cache lines, which can take two bus cycles to flush. By comparison, directly modifying the buffer in RAM can take a dozen or more bus cycles. Although utilizing a Data Cache significantly reduces the odds that a corruption may occur, corruption can still occur.

[0042] In order to overcome these problems, the present invention checks the status of the Ack packet after the present invention completes the modifications. If the Ack packet is still owned by hardware, it has not been transmitted, and the modifications were successful. If the Ack packet is owned by software, it has been transmitted, and the present invention assumes that the packet was corrupted or the modifications were ignored. In this case, the present invention simply allows the new Ack packet to be queued so that it will be transmitted at the next opportunity.

[0043] This is similar to what would happen if there was a gap in the Ack stream (which happens frequently). Here, the present invention would search the list of queued packets in transmit queue(s) 112 and find that there are no matching Ack streams to replace, so the new Ack packet must be queued.

E. Piggyback Request Support

[0044] A DOCSIS-compliant system uses a controlled-access medium for upstream transmission. A cable modem sends a request to the CMTS telling it that it has data to send, and how much time (minislots) it needs to send the data. At some point in the future, the CMTS grants time to the cable modem so that it can transmit the packet.

[0045] The request can be sent in a contention region, which is used by all cable modems, or can be sent as a piggyback request, essentially embedding the request in a packet that is currently being transmitted. As the number of cable modems on the cable plant increases, the odds that multiple cable modems will transmit in the same contention region increases, which causes the request to be lost and starts a request backoff mechanism that delays transmission of the packet.

[0046] In order to avoid collisions, it is always best to try to use the piggyback request mechanism. Additionally, using piggyback requests will often let the CMTS receive the request sooner and can let the cable modem transmit sooner, thus reducing latency.

[0047] However, if there is a single Ack stream (such as an FTP get from just one client), the Ack Filtering feature of the present invention will ensure that only one packet is ever queued. To make use of a piggyback request, a second packet must be queued.

[0048] Generally, hardware will not allow arbitrary/predictive requests to be made in software. Also, the present invention cannot just queue a dummy Ack packet, since that Ack packet might be transmitted (if a new Ack packet does not arrive soon enough).

[0049] In order to overcome this and make use of piggyback requests, the present invention actually allows the new Ack packet to be queued after performing in-place promotion. When a new Ack packet arrives, the invention searches the queue and does promotion as described above. Then, if the Ack packet that was promoted into (the old Ack packet) is at the front of

the queue, and if there are no other packets queued after it, the invention allows the new Ack packet to be queued. When the packet at the front of the queue is finally transmitted, the hardware will make a piggyback request for the Ack packet that is queued behind it. If no other Ack packets arrive, then this Ack packet is transmitted. Since it contains the same information as the Ack that was previously transmitted, there is no harm. If new Ack packets arrive, they will be promoted into the Ack packet that is now at the front of the transmit queue (noting that the new Ack packet will be queued as the new second Ack packet for piggyback purposes).

[0050] After two packets are queued, as new Ack packets arrive and are promoted into the first Ack packet, the invention must also promote them into the second Ack packet. This ensures that both packets contain identical information up to the point where the first packet is transmitted.

[0051] This causes fewer Ack packets to be dropped, since the present invention will always try to keep two packets in the queue, but the reduced latency, improved request efficiency, and reduced chance of collisions makes this worthwhile.

[0052] This applies to cases where there is a single Ack stream. If multiple Ack streams exist (such as with typical web surfing or multiple FTPs), there will rarely be just one Ack packet in the queue, and the multiple Ack packets will allow piggybacking off of each other.

F. Conclusion

[0053] FIGs. 1-4 are conceptual illustrations allowing an explanation of the present invention. It should be understood that embodiments of the present invention could be implemented in hardware, firmware, software, or a combination thereof. In such an embodiment, the various components and steps would be implemented in hardware, firmware, and/or software to perform the functions of the present invention. That is, the same piece of hardware, firmware, or module of software could perform one or more of the illustrated blocks (i.e., components or steps).

[0054] In this document, the terms "computer program medium" and "computer usable medium" are used to generally refer to media such as a removable storage unit, a hard disk installed in hard disk drive, and signals (i.e., electronic, electromagnetic, optical, or other types of signals capable of being received by a communications interface). These computer program products are means for providing software to a computer system. The invention, in an embodiment, is directed to such computer program products.

[0055] In an embodiment where aspects of the present invention is implemented using software, the software can be stored in a computer program product and loaded into computer system using a removable storage drive, hard drive, or communications interface. The control logic (software), when executed by a processor, causes the processor to perform the functions of the invention as described herein.

[0056] In another embodiment, aspects of the present invention are implemented primarily in hardware using, for example, hardware components such as an application specific integrated circuits (ASIC), a field programmable gate array (FPGA), or a similar device providing hardware assists to enable a software application to support the functions described herein. Implementation of the hardware state machine so as to perform the functions described herein will be apparent to one skilled in the relevant art(s).

[0057] In yet another embodiment, the invention is implemented using a combination of both hardware and software.

[0058] While various embodiments of the present invention have been described above, it should be understood that they have been presented by way of example, and not limitation. It will be apparent to one skilled in the relevant art(s) that various changes in form and detail can be made therein without departing from the spirit and scope of the invention. Moreover, it should be understood that the method, system, and computer program product of the present invention could be implemented in any multi-nodal communications environment governed by centralized nodes. The nodes include, but are not limited to, cable modems, set-top boxes, and headends, as well as

communication gateways, switches, routers, Internet access facilities, servers, personal computers, enhanced telephones, personal digital assistants (PDA), televisions, or the like. Thus, the present invention should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.